```
!pip install langchain-groq
!pip install langchain-openai
!pip install langchain-community
!pip install faiss-gpu
!pip install pypdf
!pip install rouge
!pip install python-dotenv
!pip install nltk
```

```
Collecting langchain-groq
  Downloading langchain_groq-0.1.9-py3-none-any.whl.metadata (2.9 kB)
Collecting groq<1,>=0.4.1 (from langchain-groq)
  Downloading groq-0.9.0-py3-none-any.whl.metadata (13 kB)
Collecting langchain-core<0.3.0,>=0.2.26 (from langchain-groq)
  Downloading langchain_core-0.2.28-py3-none-any.whl.metadata (6.2 kB)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from groq<1,>=0.4.1->langchain-groq) (3.7
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages (from groq<1,>=0.4.1->langchain-groq) (1.7.0)
Collecting httpx<1,>=0.23.0 (from groq<1,>=0.4.1->langchain-groq)
  Downloading httpx-0.27.0-py3-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from groq<1,>=0.4.1->langchain-groq) (2
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from groq<1,>=0.4.1->langchain-groq) (1.3.1)
Requirement already satisfied: typing-extensions<5,>=4.7 in /usr/local/lib/python3.10/dist-packages (from groq<1,>=0.4.1->langchain-g
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.3.0,>=0.2.26->langchain-
Collecting jsonpatch<2.0,>=1.33 (from langchain-core<0.3.0,>=0.2.26->langchain-groq)
  Downloading jsonpatch-1.33-py2.py3-none-any.whl.metadata (3.0 kB)
Collecting langsmith<0.2.0,>=0.1.75 (from langchain-core<0.3.0,>=0.2.26->langchain-groq)
  Downloading langsmith-0.1.96-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.3.0,>=0.2.26->la
Collecting tenacity!=8.4.0,<9.0.0,>=8.1.0 (from langchain-core<0.3.0,>=0.2.26->langchain-groq)
  Downloading tenacity-8.5.0-py3-none-any.whl.metadata (1.2 kB)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->groq<1,>=0.4.1->langchain-
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->groq<1,>=0.4.1->lang
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->groq<1,>=0.4.1->langchain-g
Collecting httpcore==1.* (from httpx<1,>=0.23.0->groq<1,>=0.4.1->langchain-groq)
  Downloading httpcore-1.0.5-py3-none-any.whl.metadata (20 kB)
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx<1,>=0.23.0->groq<1,>=0.4.1->langchain-groq)
  Downloading h11-0.14.0-py3-none-any.whl.metadata (8.2 kB)
Collecting jsonpointer>=1.9 (from jsonpatch<2.0,>=1.33->langchain-core<0.3.0,>=0.2.26->langchain-groq)
  Downloading jsonpointer-3.0.0-py2.py3-none-any.whl.metadata (2.3 kB)
Collecting orjson<4.0.0,>=3.9.14 (from langsmith<0.2.0,>=0.1.75->langchain-core<0.3.0,>=0.2.26->langchain-groq)
  Downloading orjson-3.10.6-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.4/50.4 kB 3.0 MB/s eta 0:00:00
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-packages (from langsmith<0.2.0,>=0.1.75->langchain-co
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->groq<1,>=0
Requirement already satisfied: pydantic-core==2.20.1 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->groq<1,>=0
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langsmith<0
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langsmith<0.2.0,>=
Downloading langchain_groq-0.1.9-py3-none-any.whl (14 kB)
Downloading groq-0.9.0-py3-none-any.whl (103 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 103.5/103.5 kB 5.5 MB/s eta 0:00:00
Downloading langchain_core-0.2.28-py3-none-any.whl (379 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 379.9/379.9 kB 19.4 MB/s eta 0:00:00
Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.6/75.6 kB 5.1 MB/s eta 0:00:00
Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 5.0 MB/s eta 0:00:00
Downloading jsonpatch-1.33-py2.py3-none-any.whl (12 kB)
Downloading langsmith-0.1.96-py3-none-any.whl (140 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 140.1/140.1 kB 9.7 MB/s eta 0:00:00
Downloading tenacity-8.5.0-py3-none-any.whl (28 kB)
Downloading jsonpointer-3.0.0-py2.py3-none-any.whl (7.6 kB)
Downloading orjson-3.10.6-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (141 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 141.1/141.1 kB 5.9 MB/s eta 0:00:00
Downloading h11-0.14.0-py3-none-any.whl (58 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 3.0 MB/s eta 0:00:00
Installing collected packages: tenacity, orjson, jsonpointer, h11, jsonpatch, httpcore, langsmith, httpx, langchain-core, groq, langc
```

```
import os
import json
from langchain_groq import ChatGroq
from langchain_openai import OpenAIEmbeddings
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains.combine_documents import create_stuff_documents_chain
from langchain_core.prompts import ChatPromptTemplate
from langchain.chains import create_retrieval_chain
from langchain_community.vectorstores import FAISS
from langchain_community.document_loaders import PyPDFDirectoryLoader

from dotenv import load_dotenv

load_dotenv()
import time


import nltk
from nltk.translate.bleu_score import sentence_bleu, corpus_bleu
from nltk.translate.meteor_score import meteor_score
from rouge import Rouge
from sklearn.metrics import precision_score, recall_score, f1_score

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
def load_groq_and_openai_keys(path):
    """Loads GROQ and OpenAI API keys from environment variables."""
    # os.environ['OPENAI_API_KEY'] = os.getenv("OPENAI_API_KEY")
    # groq_api_key = os.getenv('GROQ_API_KEY')

    f = open('/content/sample_data/environment_variable/env_var.json')

    data = json.load(f)

    groq_api_key = data['groq_api_key']
    OPENAI_API_KEY = data['OPENAI_API_KEY']

    os.environ['GROQ_API_KEY'] = groq_api_key

    os.environ['OPENAI_API_KEY'] = OPENAI_API_KEY

    return groq_api_key,OPENAI_API_KEY



def load_vectors(data_dir="/content/sample_data/us_census", max_documents=20):
    """Loads and processes documents for retrieval.

    Args:
        data_dir: Directory containing documents (default: "./us_census").
        max_documents: Maximum number of documents to load (default: 20).

    Returns:
        A tuple containing:
            embeddings: An OpenAIEmbeddings instance.
            loader: A PyPDFDirectoryLoader instance.
            final_documents: A list of processed documents.
            vectors: A FAISS vector store containing document embeddings.
    """
    embeddings = OpenAIEmbeddings()
    loader = PyPDFDirectoryLoader(data_dir)
    docs = loader.load()[:max_documents]  # Load only max_documents
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
    final_documents = text_splitter.split_documents(docs)
    vectors = FAISS.from_documents(final_documents, embeddings)
    return embeddings, loader, final_documents, vectors
```

```
def answer_question(question, groq_api_key, vectors ):
    """Answers a question using the provided GROQ API key and vector store.

    Args:
        question: The question to answer (str).
        groq_api_key: The GROQ API key (str).
        vectors: A FAISS vector store containing document embeddings.

    Returns:
        A dictionary containing the answer and context.
    """
    llm = ChatGroq(groq_api_key=groq_api_key, model_name="Llama3-8b-8192")
    prompt = ChatPromptTemplate.from_template(
        """
        Answer the questions based on the provided context only.
        Please provide the most accurate response based on the question
        <context>
        {context}
        <context>
        Questions:{input}
        """
    )
    document_chain = create_stuff_documents_chain(llm, prompt)
    retriever = vectors.as_retriever()
    retrieval_chain = create_retrieval_chain(retriever, document_chain)


    start = time.process_time()
    response = retrieval_chain.invoke({'input': question})
    print(f"Response time: {time.process_time() - start}")


    return response


from langchain.evaluation import load_evaluator
from langchain_openai import ChatOpenAI

def evaluate_response(question,generated_text,reference_text):
  """Evaluates the generated text against the reference text using BLEU, ROUGE, METEOR, Precision, Recall, and F1-score."""

  # Tokenize the texts
  generated_tokens = nltk.word_tokenize(generated_text)
  reference_tokens = nltk.word_tokenize(reference_text)

  # Calculate BLEU score
  bleu_score = sentence_bleu([reference_tokens], generated_tokens)

  # Calculate ROUGE scores
  rouge = Rouge()
  scores = rouge.get_scores(generated_text, reference_text)
  rouge_1 = scores[0]['rouge-1']['f']
  rouge_2 = scores[0]['rouge-2']['f']
  rouge_l = scores[0]['rouge-l']['f']

  #bleu, rouge_1, rouge_2, rouge_l = evaluate_response(response['answer'], reference_text)
  #print(f"BLEU: {bleu}, ROUGE-1: {rouge_1}, ROUGE-2: {rouge_2}, ROUGE-L: {rouge_l}")

  llm = ChatGroq(groq_api_key=groq_api_key, model_name="Llama3-8b-8192")
  evaluator = load_evaluator("labeled_score_string", llm=llm)
  eval_result = evaluator.evaluate_strings(
      prediction = generated_text,
      reference = reference_text,
      input=question,
  )

  print(f"Question: {question}")
  print(f"Model Prediction: {generated_text}")
  print(f"Reference Text: {reference_text}")
  #print(f"BLEU: {bleu}, ROUGE-1: {rouge_1}, ROUGE-2: {rouge_2}, ROUGE-L: {rouge_l}, Scoring_Evaluator: {eval_result['score']}")
  print(f"BLEU: {bleu}")
  print(f"ROUGE-1: {rouge_1}")
  print(f"ROUGE-2: {rouge_2}")
  print(f"ROUGE-L: {rouge_l}")
  print(f"Scoring_Evaluator: {eval_result['score']}")

  #return  bleu_score, rouge_1, rouge_2, rouge_l, eval_result
```

```
from langchain.evaluation import load_evaluator
from langchain_openai import ChatOpenAI


def find_score(question, reference_texts, response) :
  llm = ChatGroq(groq_api_key=groq_api_key, model_name="Llama3-8b-8192")
  evaluator = load_evaluator("labeled_score_string", llm=llm)
  eval_result = evaluator.evaluate_strings(
      prediction=response,
      reference=reference_texts,
      input=question,
  )
  #print(eval_result)
  return eval_result


#Test - 01

if __name__ == "__main__":
    variable_path = "/content/sample_data/environment_variable/env_var.json"

    groq_api_key,OPENAI_API_KEY = load_groq_and_openai_keys(variable_path)
    embeddings, _, _, vectors = load_vectors()

    ground_truths = "2018"
    reference_text = "2018 Survey of Income and Program Participation (SIPP)"

    question = input("Enter Your Question: ")
    response = answer_question(question, groq_api_key, vectors)
    print(f"Answer: {response['answer']}")



    '''
    # Print context snippets if desired
    print("\nAnalyze and we get in Document:")
    for i, doc in enumerate(response["context"]):
        print(doc.page_content)
        print("-------------------------------")
    '''
```

```
Enter Your Question: What is the purpose of the 2018 Survey of Income and Program Participation (SIPP) survey?
Response time: 0.16179655599999876
Answer: The purpose of the 2018 Survey of Income and Program Participation (SIPP) is to provide comprehensive information on the dynami
```

```
#Finding Model Performance:
evaluate_response(question,response['answer'],reference_text)
```

```
Question: What is the purpose of the 2018 Survey of Income and Program Participation (SIPP) survey?
Model Prediction: The purpose of the 2018 Survey of Income and Program Participation (SIPP) is to provide comprehensive information on
Reference Text: 2018 Survey of Income and Program Participation (SIPP)
BLEU: 0.2516546237169354
ROUGE-1: 0.4848484811753903
ROUGE-2: 0.39999999680000003
ROUGE-L: 0.4848484811753903
Scoring_Evaluator: 8
```

```
#Test - 02

question = "What is the joint relationship between occupation and objective characteristics?"
reference_text = "The relationship between occupation and objective characteristics likely depends on their overall relationship, which thi

response = answer_question(question, groq_api_key, vectors)
print(f"Answer: {response['answer']}")
```

```
Response time: 0.14072748899999965
Answer: The joint relationship between occupation and objective characteristics is not analyzed in this report. According to the contex
```

```
#Finding Model Performance:
evaluate_response(question,response['answer'],reference_text)
```

```
Question: What is the joint relationship between occupation and objective characteristics?
Model Prediction: The joint relationship between occupation and objective characteristics is not analyzed in this report. According to
Reference Text: The relationship between occupation and objective characteristics likely depends on their overall relationship, which t
BLEU: 0.2516546237169354
ROUGE-1: 0.44897958708871305
ROUGE-2: 0.22222221777777784
ROUGE-L: 0.3673469340274886
Scoring_Evaluator: 8
```

```
#Test-03
question = "What are the main features of employment that are highlighted in this report?"
reference_text = "The report highlights several key features of employment, including occupations, work schedules, earnings, and other job

response = answer_question(question, groq_api_key, vectors)
print(f"Answer: {response['answer']}")
```

```
Response time: 0.12637742200000446
Answer: According to the context, the main features of employment that are highlighted in this report are:

1. Occupations
2. Work schedules
3. Earnings
4. Other job characteristics

These features are highlighted using data from two surveys administered by the U.S. Census Bureau: the 2018 Survey of Income and Progra
```

```
#Finding Model Performance:
evaluate_response(question,response['answer'],reference_text)
```

```
/usr/local/lib/python3.10/dist-packages/nltk/translate/bleu_score.py:552: UserWarning:
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
Question: What are the main features of employment that are highlighted in this report?
Model Prediction: According to the context, the main features of employment that are highlighted in this report are:

1. Occupations
2. Work schedules
3. Earnings
4. Other job characteristics

These features are highlighted using data from two surveys administered by the U.S. Census Bureau: the 2018 Survey of Income and Progra
Reference Text: The report highlights several key features of employment, including occupations, work schedules, earnings, and other jo
BLEU: 0.2516546237169354
ROUGE-1: 0.17910447382490538
ROUGE-2: 0.05405405066471898
ROUGE-L: 0.14925372755624872
Scoring_Evaluator: 9
```